WHY DO WE NEED DEEP GENERATIVE MODELING?

Jakub M. Tomczak

13 January 2020

VRIJE UNIVERSITEIT AMSTERDAM

Introduction

IS GENERATIVE MODELING IMPORTANT?

We learn a neural network to classify images:

IS GENERATIVE MODELING IMPORTANT?

We learn a neural network to classify images:

IS GENERATIVE MODELING IMPORTANT?

We learn a neural network to classify images:

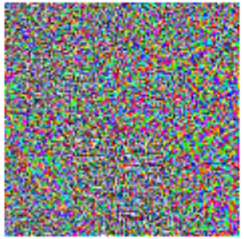
p(**panda**|x)=0.99

...

IS GENERATIVE MODELING IMPORTANT?

We learn a neural network to classify images:

+



=

p(**panda**|x)=0.99

...

noise

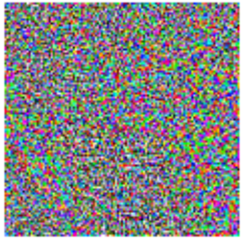
IS GENERATIVE MODELING IMPORTANT?

We learn a neural network to classify images:

p(**panda**|x)=0.99

...

+



noise

=

p(**panda**|x)=0.01

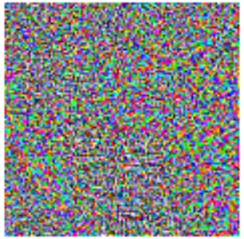
...

p(**dog**|x)=0.9

IS GENERATIVE MODELING IMPORTANT?

We learn a neural network to classify images:

+



=

p(**panda**|x)=0.99
...

noise

p(**panda**|x)=0.01
...
p(**dog**|x)=0.9

There is no semantic understanding of images.

IS GENERATIVE MODELING IMPORTANT?

This simple example shows that:

- A discriminative model is (probably) **not enough**.
- We need a notion of **uncertainty**.
- We need to **understand** the reality.

IS GENERATIVE MODELING IMPORTANT?

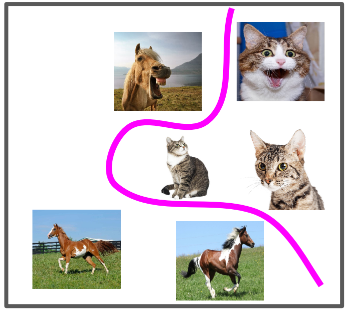
This simple example shows that:

- A discriminative model is (probably) **not enough**.
- We need a notion of **uncertainty**.
- We need to **understand** the reality.

A possible solution is **generative modeling**.

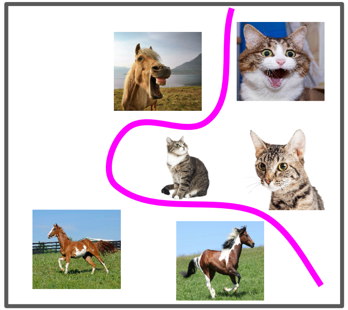
IS GENERATIVE MODELING IMPORTANT?

IS GENERATIVE MODELING IMPORTANT?

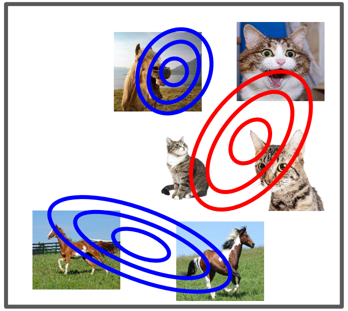


$$p_\theta(y|x)$$

IS GENERATIVE MODELING IMPORTANT?

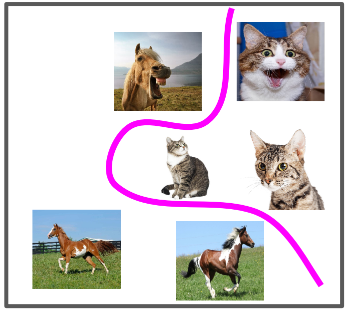


$p_\theta(y|x)$

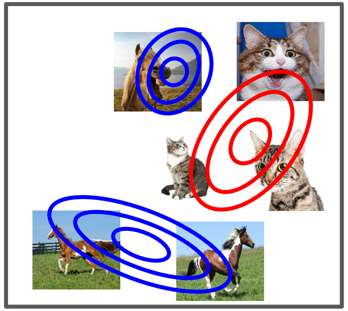


$p_\theta(x, y) = p_\theta(y|x)\ p_\theta(x)$

IS GENERATIVE MODELING IMPORTANT?

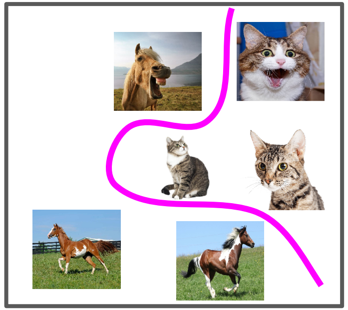


$$p_\theta(y|x)$$



$$p_\theta(x, y) = p_\theta(y|x)\ p_\theta(x)$$

IS GENERATIVE MODELING IMPORTANT?

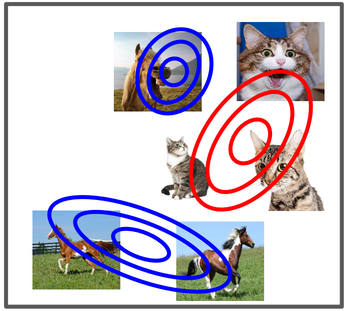


$$p_\theta(y|x)$$

High probability of a **horse.**

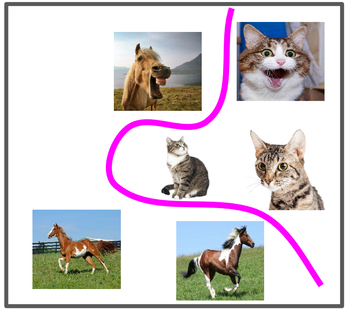
=

Highly probable decision!



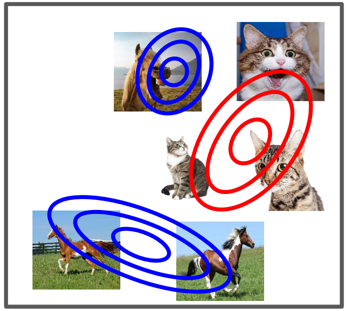
$$p_\theta(x, y) = p_\theta(y|x)\ p_\theta(x)$$

IS GENERATIVE MODELING IMPORTANT?



$$p_\theta(y|x)$$

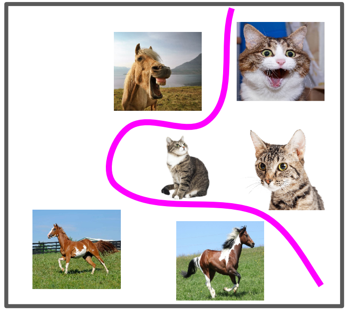
High probability of a **horse.**
=
Highly probable decision!



$$p_\theta(x, y) = p_\theta(y|x)\ p_\theta(x)$$

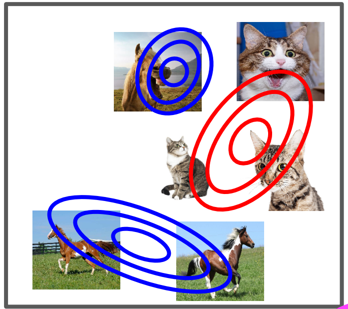
High probability of a **horse.**
x
Low probability of the **object**
=
Uncertain decision!

IS GENERATIVE MODELING IMPORTANT?



$$p_\theta(y|x)$$

High probability of a **horse.**
=
Highly probable decision!



$$p_\theta(x,y) = p_\theta(y|x)\, p_\theta(x)$$

!

High probability of a **horse.**
x
Low probability of the **object**
=
Uncertain decision!

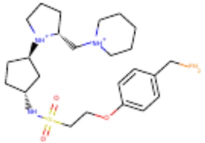
WHERE DO WE USE DEEP GENERATIVE MODELING?

" i want to talk to you . "
"i want to be with you . "
"i do n't want to be with you . "
i do n't want to be with you .
she did n't want to be with him .

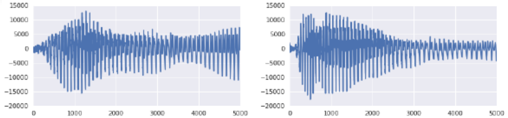
he was silent for a long moment .
he was silent for a moment .
it was quiet for a moment .
it was dark and cold .
there was a pause .
it was my turn .

Text analysis

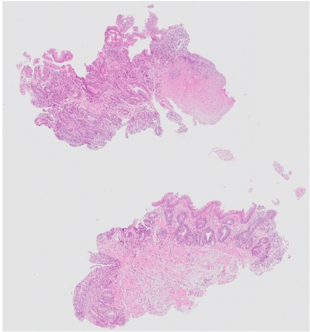
Image analysis



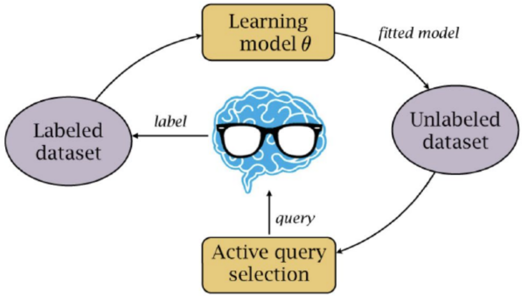
Graph analysis



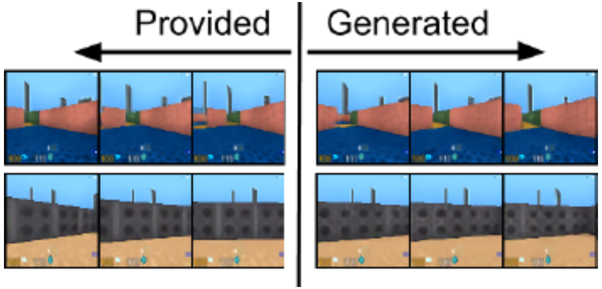
Audio analysis



Medical data



Active Learning



Reinforcement Learning

and more...

VU

HOW TO FORMULATE GENERATIVE MODELS?

Generative model

- **Autoregressive (*e.g.*, PixelCNN)**
- **Flow-based (e.g., RealNVP, GLOW)**
- **Latent variable models**
  - **Implicit models (*e.g.*, GANs)**
  - **Prescribed models (*e.g.*, VAE)**

HOW TO FORMULATE GENERATIVE MODELS?

	Training	Likelihood	Sampling	Compression
Autoregressive models (e.g., PixelCNN)	Stable	Exact	Slow	No
Flow-based models (e.g., RealNVP)	Stable	Exact	Fast/Slow	No
Implicit models (e.g., GANs)	Unstable	No	Fast	No
Prescribed models (e.g., VAEs)	Stable	Approximate	Fast	Yes

HOW TO FORMULATE GENERATIVE MODELS?

	Training	Likelihood	Sampling	Compression
Autoregressive models (e.g., PixelCNN)	Stable	Exact	Slow	No
Flow-based models (e.g., RealNVP)	Stable	Exact	Fast/Slow	No
Implicit models (e.g., GANs)	Unstable	No	Fast	No
Prescribed models (e.g., VAEs)	Stable	Approximate	Fast	Yes

GENERATIVE MODELS AS (SPHERICAL) COWS

GENERATIVE MODELS AS (SPHERICAL) COWS

flow-based models

GENERATIVE MODELS AS (SPHERICAL) COWS

flow-based models

latent variable models

Deep latent variable models

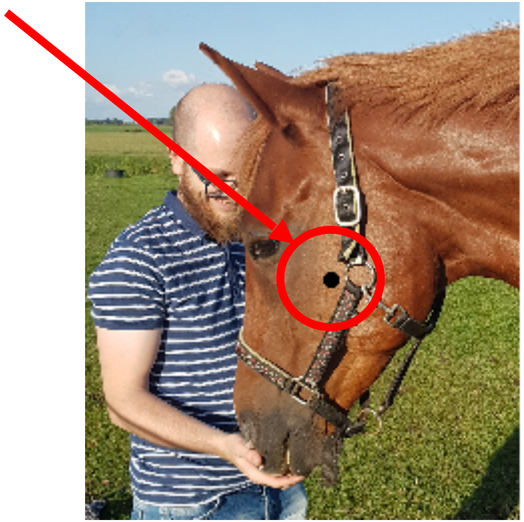
VRIJE UNIVERSITEIT AMSTERDAM

GENERATIVE MODELING

Modeling in high-dimensional spaces is difficult.

GENERATIVE MODELING

Modeling in high-dimensional spaces is difficult.



GENERATIVE MODELING

Modeling in high-dimensional spaces is difficult.

Modeling **all dependencies** among pixels:

$$p(\mathbf{x}) = \frac{1}{Z} \prod_{c=1}^{C} \psi_c(\mathbf{x}_c)$$

GENERATIVE MODELING

Modeling in high-dimensional spaces is difficult.

Modeling **all dependencies** among pixels:

$$p(\mathbf{x}) = \frac{1}{Z} \prod_{c=1}^{C} \psi_c(\mathbf{x}_c)$$

problematic

GENERATIVE MODELING

Modeling in high-dimensional spaces is difficult.

Modeling **all dependencies** among pixels:

$$p(\mathbf{x}) = \frac{1}{Z} \prod_{c=1}^{C} \psi_c(\mathbf{x}_c)$$

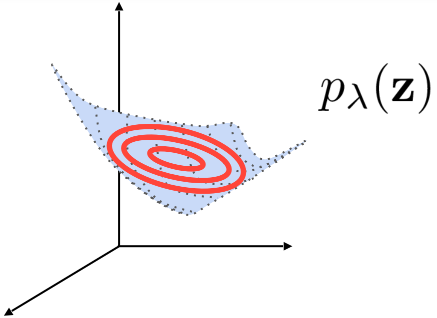
problematic

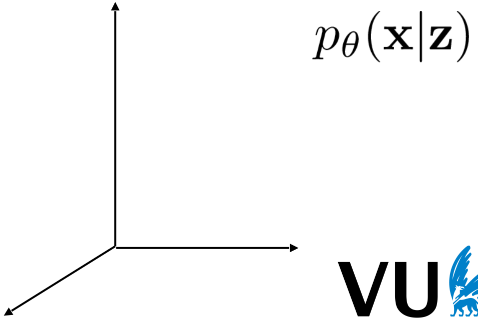
A possible **solution**: **Latent Variable Models**!

GENERATIVE MODELING WITH LATENT VARIABLES

Generative process:

1. $\mathbf{z} \sim p_\lambda(\mathbf{z})$
2. $\mathbf{x} \sim p_\theta(\mathbf{x}|\mathbf{z})$

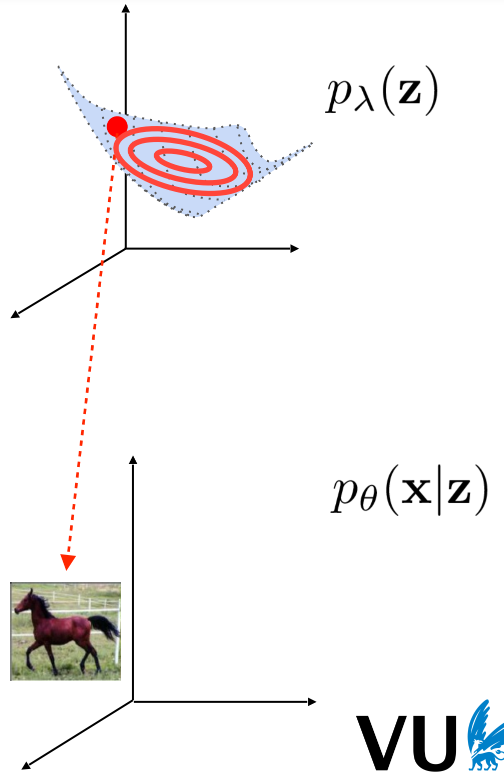




GENERATIVE MODELING WITH LATENT VARIABLES

Generative process:

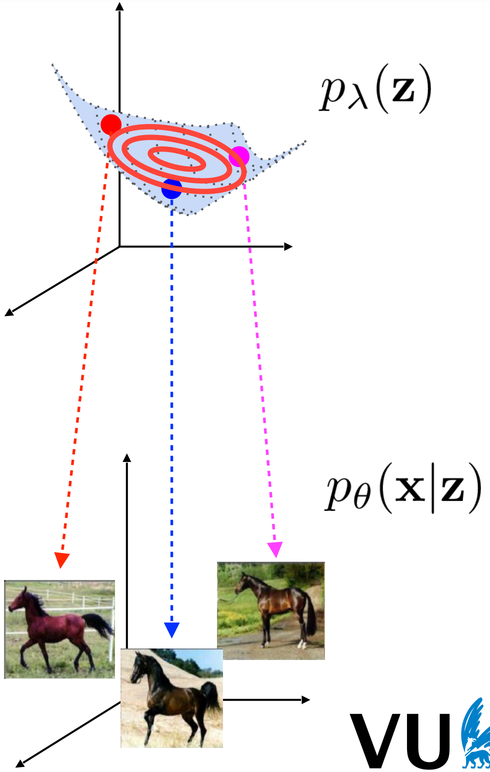
1. $\mathbf{z} \sim p_\lambda(\mathbf{z})$
2. $\mathbf{x} \sim p_\theta(\mathbf{x}|\mathbf{z})$



GENERATIVE MODELING WITH LATENT VARIABLES

Generative process:

1. $\mathbf{z} \sim p_\lambda(\mathbf{z})$
2. $\mathbf{x} \sim p_\theta(\mathbf{x}|\mathbf{z})$



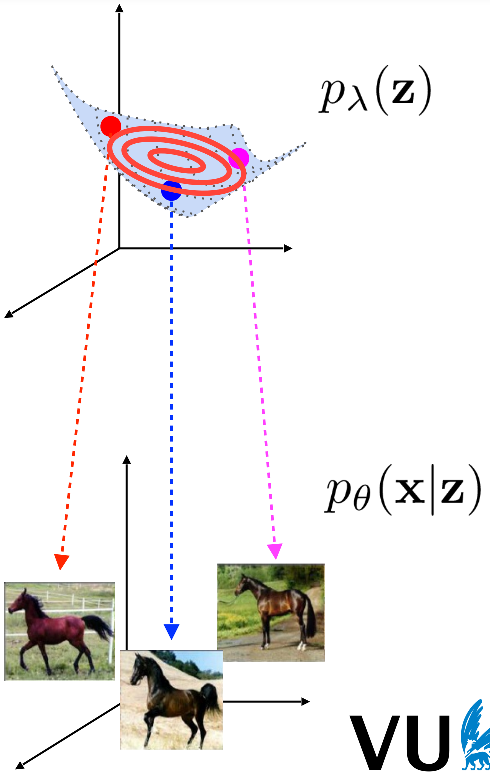
GENERATIVE MODELING WITH LATENT VARIABLES

Generative process:

1. $\mathbf{z} \sim p_{\lambda}(\mathbf{z})$
2. $\mathbf{x} \sim p_{\theta}(\mathbf{x}|\mathbf{z})$

Log of marginal distribution:

$$\log p_{\vartheta}(\mathbf{x}) = \log \int p_{\theta}(\mathbf{x}|\mathbf{z})\ p_{\lambda}(\mathbf{z})\mathrm{d}\mathbf{z}$$



GENERATIVE MODELING WITH LATENT VARIABLES

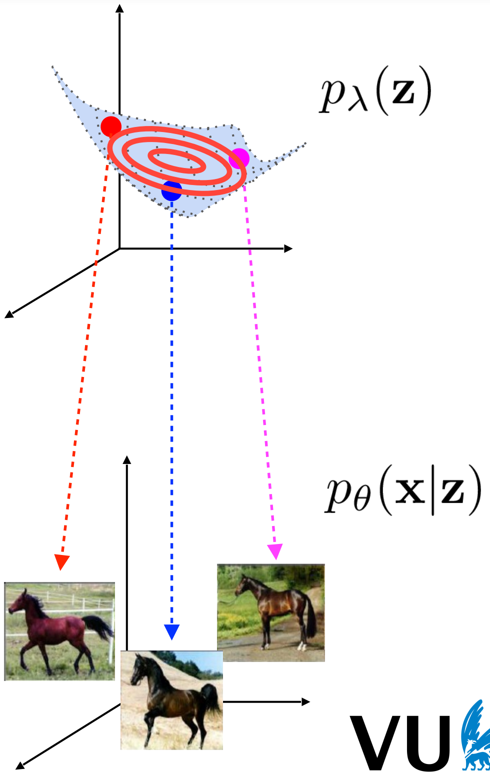
Generative process:

$$1.\ \mathbf{z} \sim p_\lambda(\mathbf{z})$$
$$2.\ \mathbf{x} \sim p_\theta(\mathbf{x}|\mathbf{z})$$

Log of marginal distribution:

$$\log p_\vartheta(\mathbf{x}) = \log \int p_\theta(\mathbf{x}|\mathbf{z})\ p_\lambda(\mathbf{z}) \mathrm{d}\mathbf{z}$$

How to train such model efficiently?



VARIATIONAL INFERENCE FOR LATENT VARIABLE MODELS

$$
\begin{aligned}
\log p_\vartheta(\mathbf{x}) &= \log \int p_\theta(\mathbf{x}|\mathbf{z})\ p_\lambda(\mathbf{z})d\mathbf{z} \\
&= \log \int \frac{q_\phi(\mathbf{z}|\mathbf{x})}{q_\phi(\mathbf{z}|\mathbf{x})} p_\theta(\mathbf{x}|\mathbf{z})\ p_\lambda(\mathbf{z})d\mathbf{z} \\
&\geq \int q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{p_\theta(\mathbf{x}|\mathbf{z})\ p_\lambda(\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} d\mathbf{z} \\
&= \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\Big[\log p_\theta(\mathbf{x}|\mathbf{z})\Big] - \mathrm{KL}\Big(q_\phi(\mathbf{z}|\mathbf{x}) || p_\lambda(\mathbf{z})\Big)
\end{aligned}
$$

VARIATIONAL INFERENCE FOR LATENT VARIABLE MODELS

$$
\begin{aligned}
\log p_\vartheta(\mathbf{x}) &= \log \int p_\theta(\mathbf{x}|\mathbf{z})\ p_\lambda(\mathbf{z})\mathrm{d}\mathbf{z} \\
&= \log \int \frac{q_\phi(\mathbf{z}|\mathbf{x})}{q_\phi(\mathbf{z}|\mathbf{x})} p_\theta(\mathbf{x}|\mathbf{z})\ p_\lambda(\mathbf{z})\mathrm{d}\mathbf{z} \\
&\geq \int q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{p_\theta(\mathbf{x}|\mathbf{z})\ p_\lambda(\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} \mathrm{d}\mathbf{z} \\
&= \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\Big[\log p_\theta(\mathbf{x}|\mathbf{z})\Big] - \mathrm{KL}\Big(q_\phi(\mathbf{z}|\mathbf{x})||p_\lambda(\mathbf{z})\Big)
\end{aligned}
$$

Variational posterior

VARIATIONAL INFERENCE FOR LATENT VARIABLE MODELS

$$\log p_{\vartheta}(\mathbf{x}) = \log \int p_{\theta}(\mathbf{x}|\mathbf{z})\ p_{\lambda}(\mathbf{z})\mathrm{d}\mathbf{z}$$

$$= \log \int \frac{q_{\phi}(\mathbf{z}|\mathbf{x})}{q_{\phi}(\mathbf{z}|\mathbf{x})} p_{\theta}(\mathbf{x}|\mathbf{z})\ p_{\lambda}(\mathbf{z})\mathrm{d}\mathbf{z}$$

Jensen's inequality

$$\geq \int q_{\phi}(\mathbf{z}|\mathbf{x}) \log \frac{p_{\theta}(\mathbf{x}|\mathbf{z})\ p_{\lambda}(\mathbf{z})}{q_{\phi}(\mathbf{z}|\mathbf{x})} \mathrm{d}\mathbf{z}$$

$$= \mathbb{E}_{\mathbf{z}\sim q_{\phi}(\mathbf{z}|\mathbf{x})}\Big[\log p_{\theta}(\mathbf{x}|\mathbf{z})\Big] - \mathrm{KL}\Big(q_{\phi}(\mathbf{z}|\mathbf{x})||p_{\lambda}(\mathbf{z})\Big)$$

VARIATIONAL INFERENCE FOR LATENT VARIABLE MODELS

$$
\begin{aligned}
\log p_{\vartheta}(\mathbf{x}) &= \log \int p_{\theta}(\mathbf{x}|\mathbf{z})\ p_{\lambda}(\mathbf{z}) \mathrm{d}\mathbf{z} \\
&= \log \int \frac{q_{\phi}(\mathbf{z}|\mathbf{x})}{q_{\phi}(\mathbf{z}|\mathbf{x})} p_{\theta}(\mathbf{x}|\mathbf{z})\ p_{\lambda}(\mathbf{z}) \mathrm{d}\mathbf{z} \\
&\geq \int q_{\phi}(\mathbf{z}|\mathbf{x}) \log \frac{p_{\theta}(\mathbf{x}|\mathbf{z})\ p_{\lambda}(\mathbf{z})}{q_{\phi}(\mathbf{z}|\mathbf{x})} \mathrm{d}\mathbf{z} \\
&= \underbrace{\mathbb{E}_{\mathbf{z}\sim q_{\phi}(\mathbf{z}|\mathbf{x})}\Big[\log p_{\theta}(\mathbf{x}|\mathbf{z})\Big]}_{\text{Reconstruction error}} - \underbrace{\mathrm{KL}\Big(q_{\phi}(\mathbf{z}|\mathbf{x})||p_{\lambda}(\mathbf{z})\Big)}_{\text{Regularization}}
\end{aligned}
$$

VARIATIONAL INFERENCE FOR LATENT VARIABLE MODELS

decoder

encoder

marginal

$$
\begin{aligned}
\log p_\vartheta(\mathbf{x}) &= \log \int p_\theta(\mathbf{x}|\mathbf{z})\; p_\lambda(\mathbf{z}) \mathrm{d}\mathbf{z} \\
&= \log \int \frac{q_\phi(\mathbf{z}|\mathbf{x})}{q_\phi(\mathbf{z}|\mathbf{x})} p_\theta(\mathbf{x}|\mathbf{z})\; p_\lambda(\mathbf{z}) \mathrm{d}\mathbf{z} \\
&\geq \int q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{p_\theta(\mathbf{x}|\mathbf{z})\; p_\lambda(\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} \mathrm{d}\mathbf{z} \\
&= \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}|\mathbf{z})\right] - \mathrm{KL}\left(q_\phi(\mathbf{z}|\mathbf{x}) || p_\lambda(\mathbf{z})\right)
\end{aligned}
$$

VARIATIONAL INFERENCE FOR LATENT VARIABLE MODELS

$$
\begin{aligned}
\log p_\vartheta(\mathbf{x}) &= \log \int p_\theta(\mathbf{x}|\mathbf{z})\ p_\lambda(\mathbf{z}) d\mathbf{z} \\
&= \log \int \frac{q_\phi(\mathbf{z}|\mathbf{x})}{q_\phi(\mathbf{z}|\mathbf{x})} p_\theta(\mathbf{x}|\mathbf{z})\ p_\lambda(\mathbf{z}) d\mathbf{z} \\
&\geq \int q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{p_\theta(\mathbf{x}|\mathbf{z})\ p_\lambda(\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} d\mathbf{z} \\
&= \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}|\mathbf{z})\right] - \mathrm{KL}\left(q_\phi(\mathbf{z}|\mathbf{x}) || p_\lambda(\mathbf{z})\right)
\end{aligned}
$$

decoder

encoder

marginal

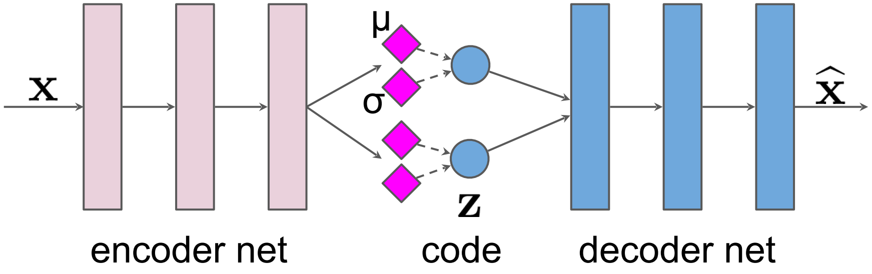
= Variational Auto-Encoder

VU

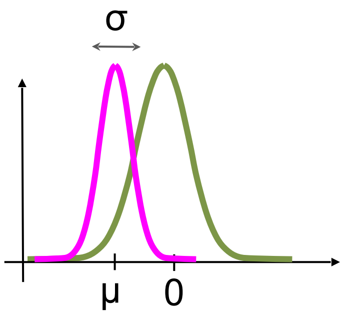
VARIATIONAL AUTO-ENCODERS

Variational posterior (**encoder**) and likelihood function (**decoder**) are parameterized by neural networks.

Reparameterization trick: move the stochasticity to independent random variables



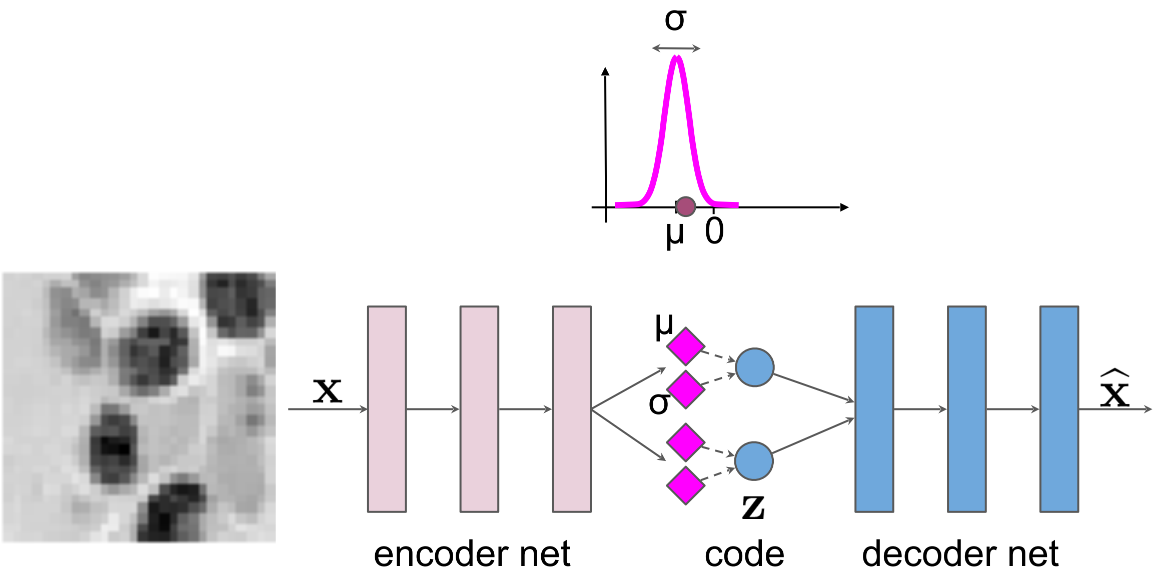
$$\mathbf{z} = f(\boldsymbol{\mu}, \boldsymbol{\sigma}; \boldsymbol{\varepsilon}), \text{ where } \boldsymbol{\varepsilon} \sim p(\boldsymbol{\varepsilon})$$



VARIATIONAL AUTO-ENCODERS

VAE copies input to output through a **bottleneck**.

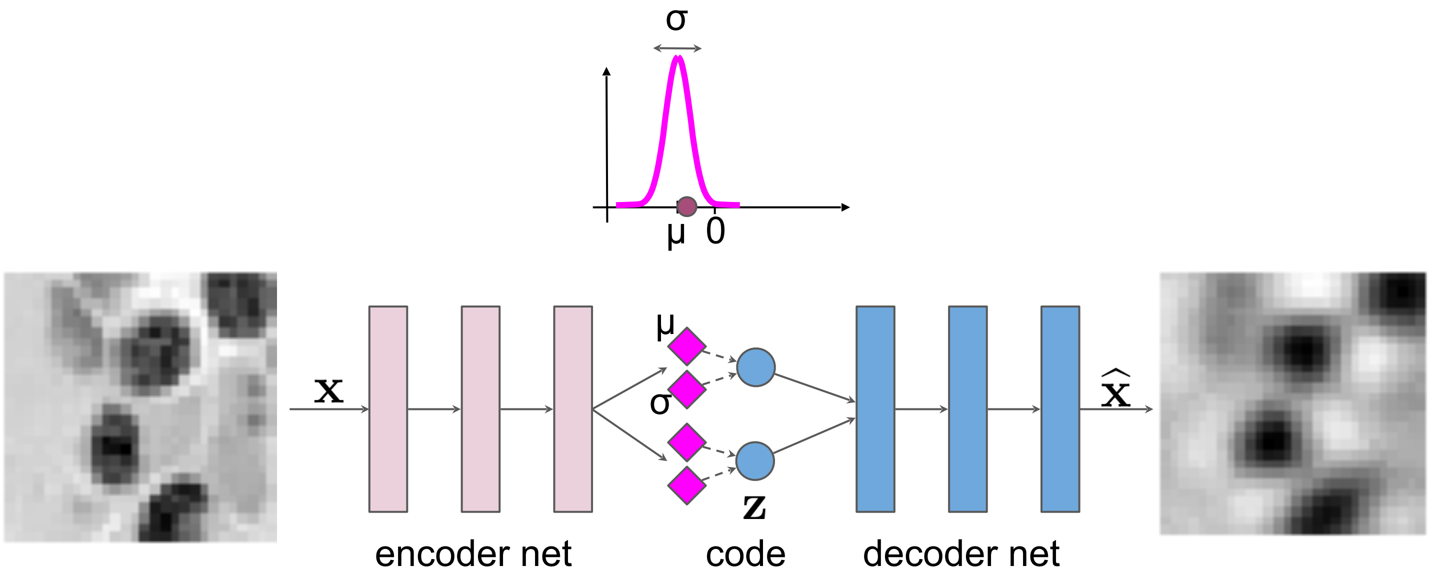
VAE learns a **code** of the data.



VARIATIONAL AUTO-ENCODERS

VAE copies input to output through a **bottleneck**.

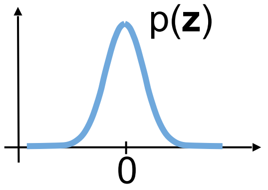
VAE learns a **code** of the data.

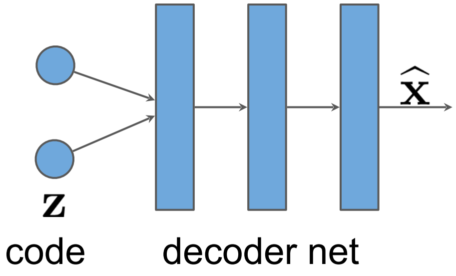


VARIATIONAL AUTO-ENCODERS

VAE has a **marginal** on the latent code.

VAE can **generate** new data.

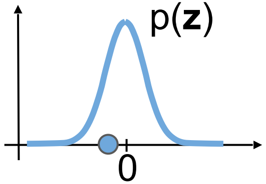


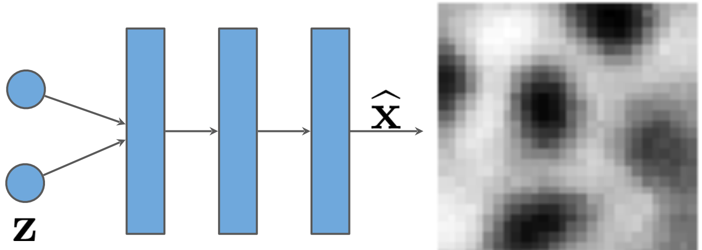


VARIATIONAL AUTO-ENCODERS

VAE has a **marginal** on the latent code.

VAE can **generate** new data.



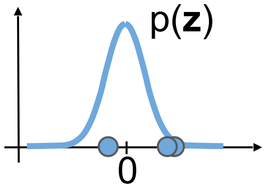


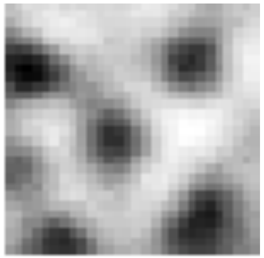
code decoder net

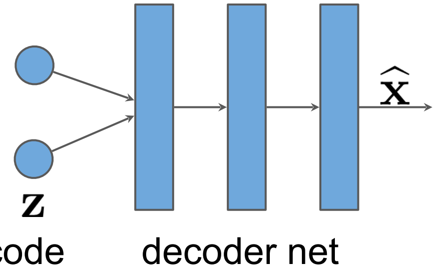
VARIATIONAL AUTO-ENCODERS

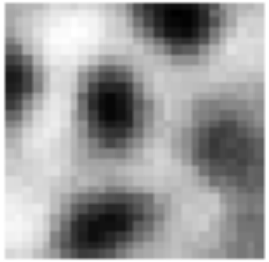
VAE has a **marginal** on the latent code.

VAE can **generate** new data.









COMMON ISSUES WITH VAES

$$q_\phi(\mathbf{z}|\mathbf{x}) \propto p_\theta(\mathbf{x}|\mathbf{z})\ p_\lambda(\mathbf{z})$$

Weak decoders → bad generations/reconstructions

Weak encoders → bad latent representation

Weak marginals → bad generations

Variational posteriors → what family of distributions?

Others…

COMPONENTS OF VAES

$$q_{\phi}(\mathbf{z}|\mathbf{x}) \propto p_{\theta}(\mathbf{x}|\mathbf{z})\ p_{\lambda}(\mathbf{z})$$

Resnets
DRAW
Autoregressive models
Normalizing flows

Autoregressive models
Normalizing flows
VampPrior
Implicit prior

COMPONENTS OF VAES

$$q_{\phi}(\mathbf{z}|\mathbf{x}) \propto p_{\theta}(\mathbf{x}|\mathbf{z})\ p_{\lambda}(\mathbf{z})$$

Normalizing flows
Discrete encoders
Hyperspherical dist.
Hyperbolic-normal dist.
Group theory

Resnets
DRAW
Autoregressive models
Normalizing flows

Autoregressive models
Normalizing flows
VampPrior
Implicit prior

COMPONENTS OF VAES

$$q_\phi(\mathbf{z}|\mathbf{x}) \propto p_\theta(\mathbf{x}|\mathbf{z})\ p_\lambda(\mathbf{z})$$

Normalizing flows
Discrete encoders
Hyperspherical dist.
Hyperbolic-normal dist.
Group theory

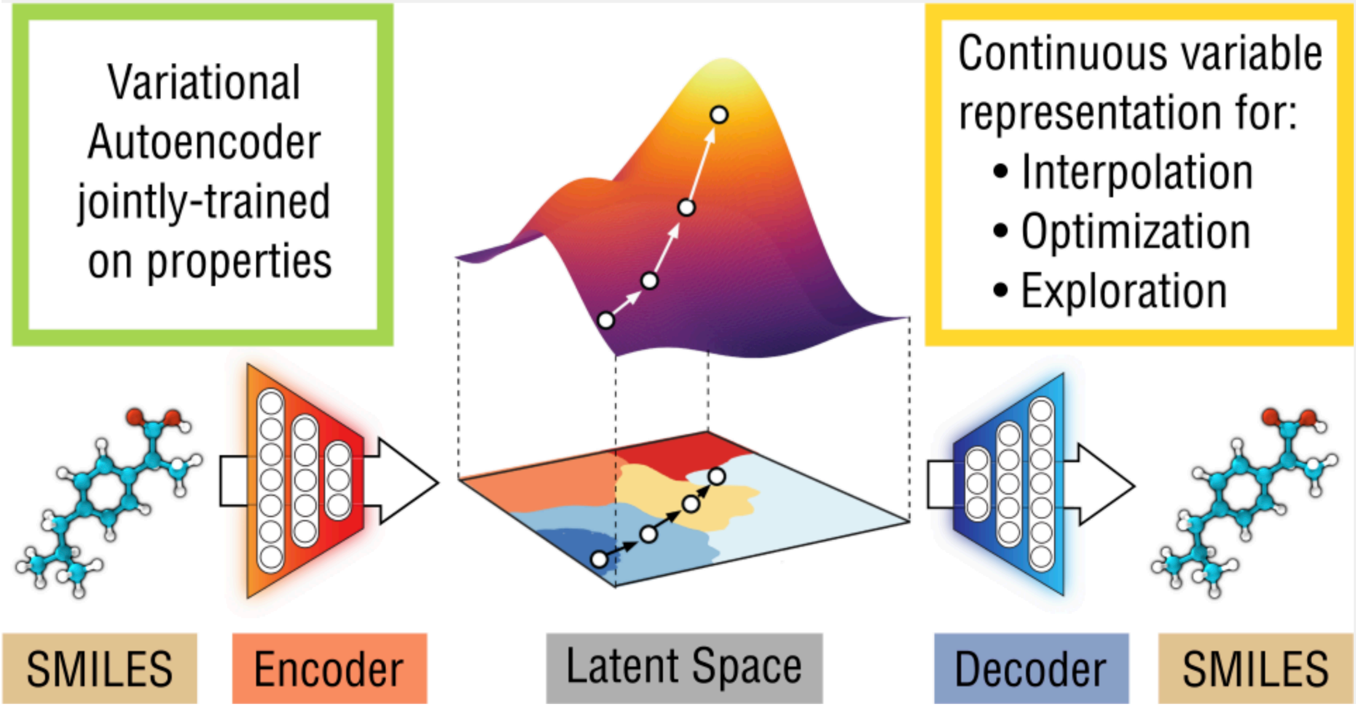
Resnets
DRAW
Autoregressive models
Normalizing flows

Autoregressive models
Normalizing flows
VampPrior
Implicit prior

$$\mathrm{ELBO}(\mathbf{x};\theta,\phi,\lambda)$$

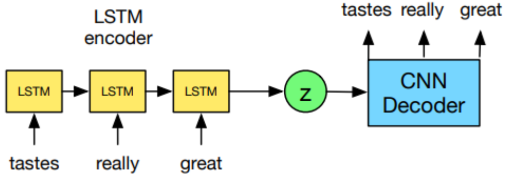
Adversarial learning
MMD
Wasserstein AE

APPLICATIONS

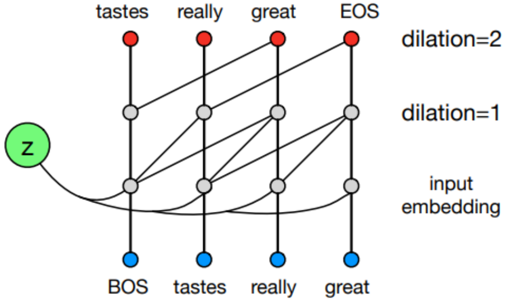


Gómez-Bombarelli, Rafael, et al. "Automatic chemical design using a data-driven continuous representation of molecules." *ACS central science* 4.2 (2018)

APPLICATIONS

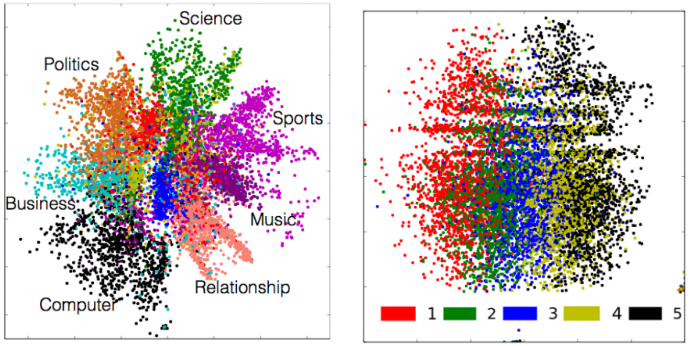


(a) VAE training graph using a dilated CNN decoder.



(b) Digram of dilated CNN decoder.

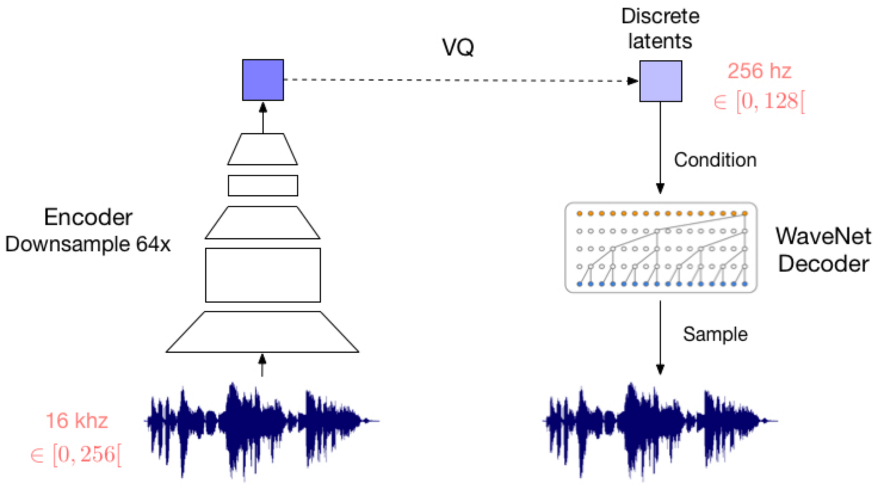
1 star	the food was good but the service was horrible . took forever to get our food . we had to ask twice for our check after we got our food . will not return .
2 star	the food was good , but the service was terrible . took forever to get someone to take our drink order . had to ask 3 times to get the check . food was ok , nothing to write about .
3 star	came here for the first time last night . food was good . service was a little slow . food was just ok .
4 star	food was good , service was a little slow , but the food was pretty good . i had the grilled chicken sandwich and it was really good . will definitely be back !
5 star	food was very good , service was fast and friendly . food was very good as well . will be back !



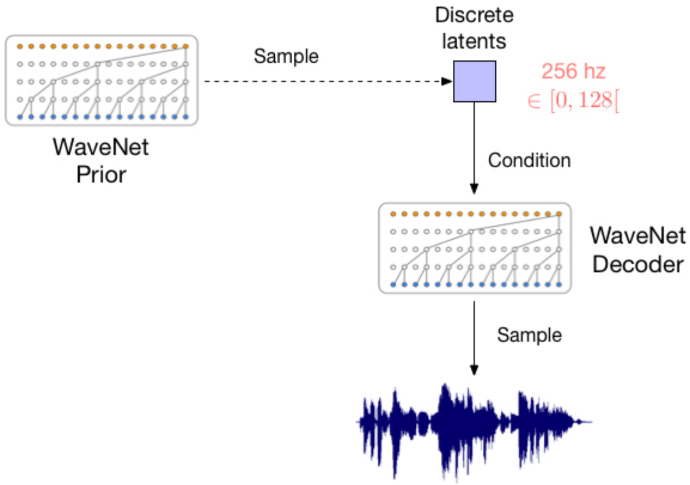
(a) Yahoo (b) Yelp

Yang, Z., Hu, Z., Salakhutdinov, R., & Berg-Kirkpatrick, T. (2017). Improved variational autoencoders for text modeling using dilated convolutions. *ICML 2017*

APPLICATIONS



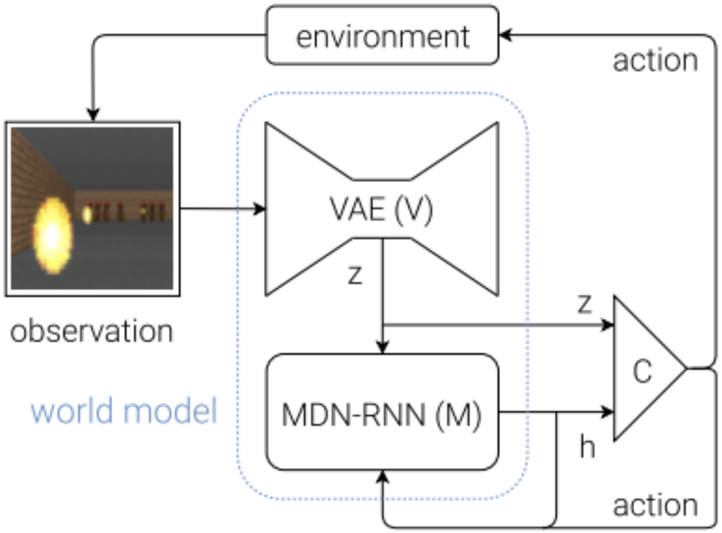
reconstruction

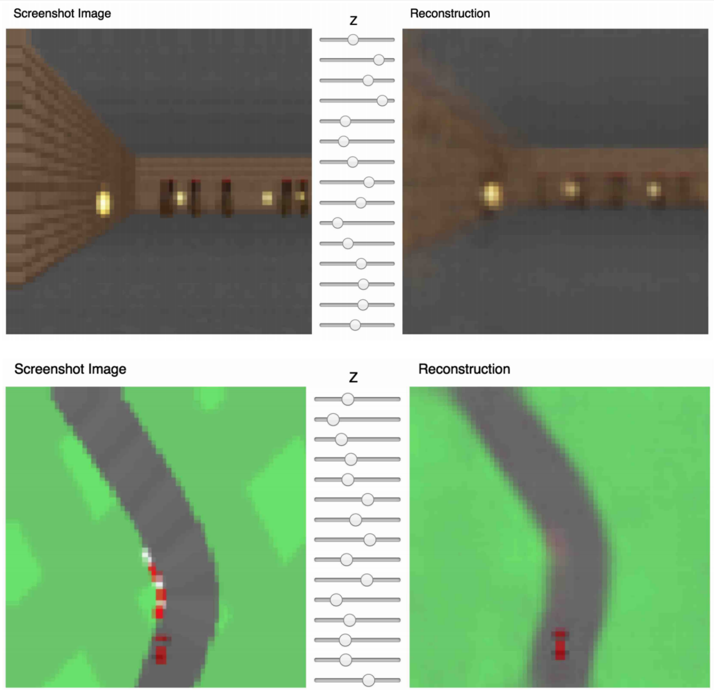


generation

van den Oord, A., & Vinyals, O. (2017). Neural discrete representation learning. *NIPS 2017*.

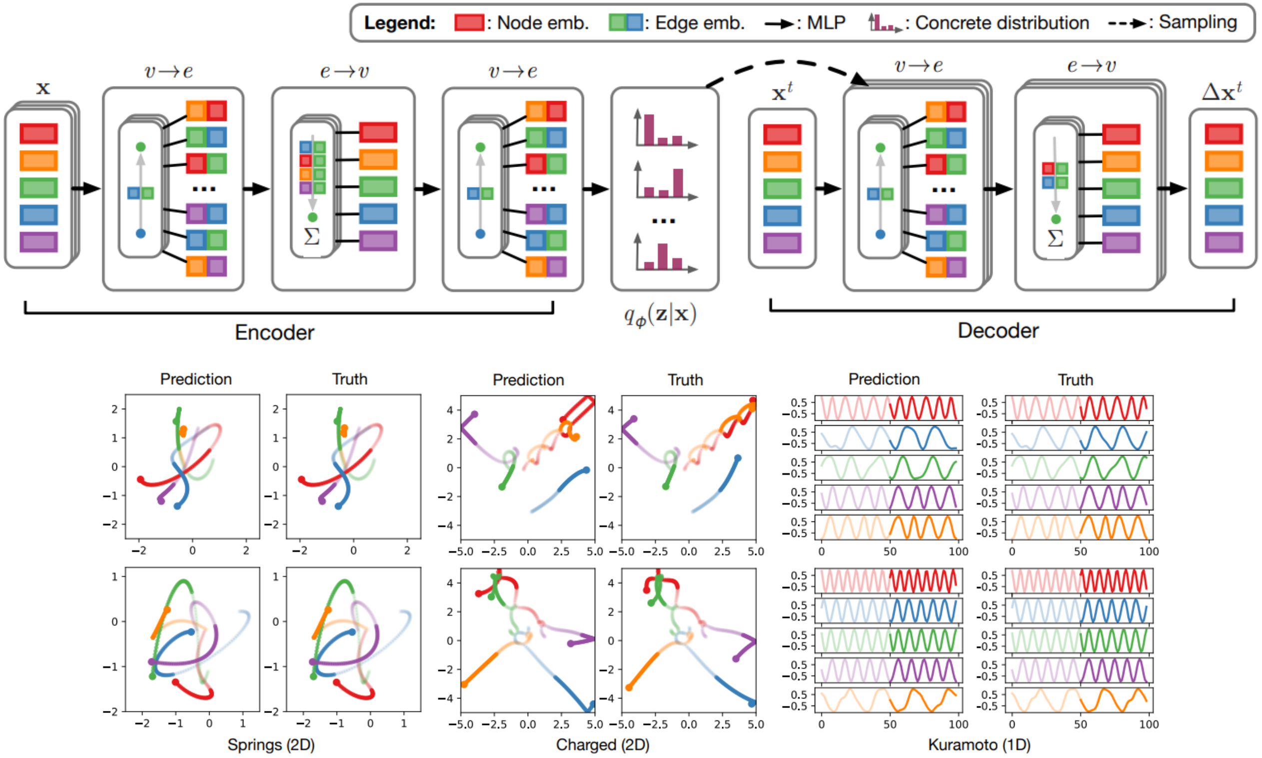
APPLICATIONS





Ha, D., & Schmidhuber, J. (2018). World models. arXiv preprint. *arXiv preprint arXiv:1803.10122*.

APPLICATIONS



Kipf, T., Fetaya, E., Wang, K. C., Welling, M., & Zemel, R. (2018). Neural relational inference for interacting systems. *ICML 2018*.

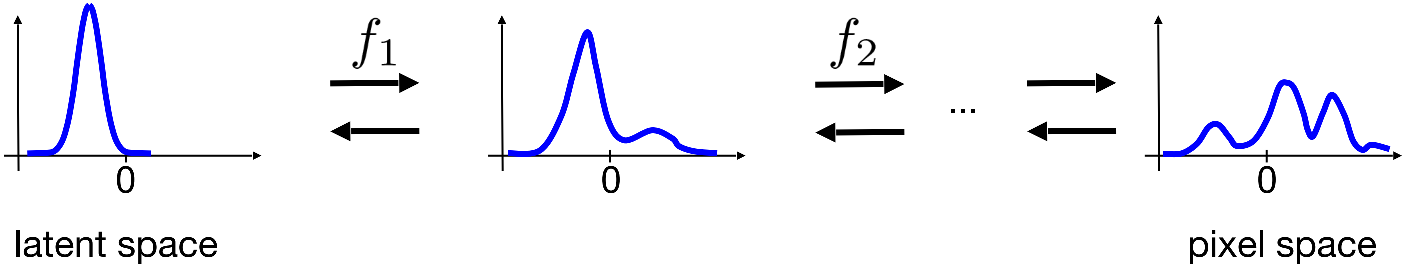
Flow-based models

THE CHANGE OF VARIABLES FORMULA

Let's recall the change of variables formula with invertible transformations:

$$p(\mathbf{x}) = \pi_0\left(\mathbf{z}_0\right) \prod_{i=1}^{K} \left| \det \frac{\partial f_i(\mathbf{z}_{i-1})}{\partial \mathbf{z}_{i-1}} \right|^{-1}$$

We can think of it as an invertible neural network:



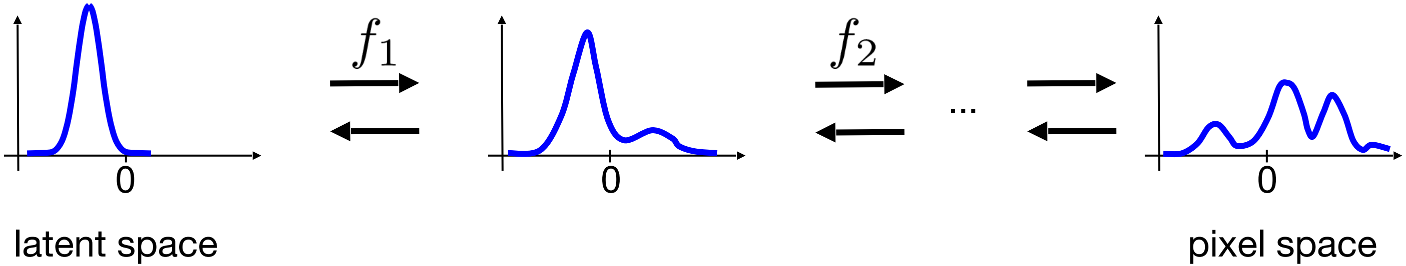
Rippel, O., & Adams, R. P. (2013). High-dimensional probability estimation with deep density models. arXiv preprint arXiv:1302.5125.

THE CHANGE OF VARIABLES FORMULA

Let's recall the change of variables formula with invertible transformations:

$$p(\mathbf{x}) = \pi_0\left(\mathbf{z}_0\right) \prod_{i=1}^{K} \left|\det \frac{\partial f_i(\mathbf{z}_{i-1})}{\partial \mathbf{z}_{i-1}}\right|^{-1}$$

We can think of it as an invertible neural network:



Rippel, O., & Adams, R. P. (2013). High-dimensional probability estimation with deep density models. arXiv preprint arXiv:1302.5125.

REALNVP

Design the invertible transformations as follows:

$$\mathbf{y}_{1:d} = \mathbf{x}_{1:d}$$

$$\mathbf{y}_{d+1:D} = \mathbf{x}_{d+1:D} \odot \exp\left(s\left(\mathbf{x}_{1:d}\right)\right) + t\left(\mathbf{x}_{1:d}\right)$$

Dinh, L., Sohl-Dickstein, J., & Bengio, S. (2016). Density estimation using real nvp. arXiv preprint arXiv:1605.08803.

REALNVP

Design the invertible transformations as follows:

$$\mathbf{y}_{1:d} = \mathbf{x}_{1:d}$$

$$\mathbf{y}_{d+1:D} = \mathbf{x}_{d+1:D} \odot \exp\left(s\left(\mathbf{x}_{1:d}\right)\right) + t\left(\mathbf{x}_{1:d}\right)$$

Invertible by design:

$$\begin{cases} \mathbf{y}_{1:d} & = \mathbf{x}_{1:d} \\ \mathbf{y}_{d+1:D} & = \mathbf{x}_{d+1:D} \odot \exp\left(s\left(\mathbf{x}_{1:d}\right)\right) + t\left(\mathbf{x}_{1:d}\right) \end{cases} \Leftrightarrow \begin{cases} \mathbf{x}_{1:d} & = \mathbf{y}_{1:d} \\ \mathbf{x}_{d+1:D} & = \left(\mathbf{y}_{d+1:D} - t\left(\mathbf{y}_{1:d}\right)\right) \odot \exp\left(-s\left(\mathbf{y}_{1:d}\right)\right) \end{cases}$$

Dinh, L., Sohl-Dickstein, J., & Bengio, S. (2016). Density estimation using real nvp. arXiv preprint arXiv:1605.08803.

REALNVP

Design the invertible transformations as follows:

$$\mathbf{y}_{1:d} = \mathbf{x}_{1:d}$$

$$\mathbf{y}_{d+1:D} = \mathbf{x}_{d+1:D} \odot \exp\left(s\left(\mathbf{x}_{1:d}\right)\right) + t\left(\mathbf{x}_{1:d}\right)$$

Invertible by design:

$$\begin{cases} \mathbf{y}_{1:d} & = \mathbf{x}_{1:d} \\ \mathbf{y}_{d+1:D} & = \mathbf{x}_{d+1:D} \odot \exp\left(s\left(\mathbf{x}_{1:d}\right)\right) + t\left(\mathbf{x}_{1:d}\right) \end{cases} \Leftrightarrow \begin{cases} \mathbf{x}_{1:d} & = \mathbf{y}_{1:d} \\ \mathbf{x}_{d+1:D} & = \left(\mathbf{y}_{d+1:D} - t\left(\mathbf{y}_{1:d}\right)\right) \odot \exp\left(-s\left(\mathbf{y}_{1:d}\right)\right) \end{cases}$$

Easy Jacobian:

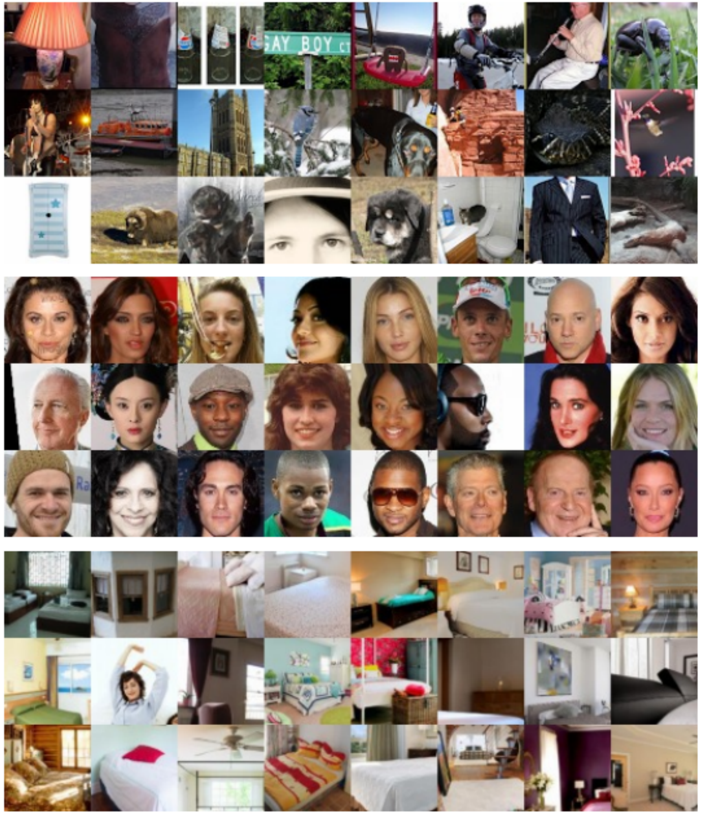
$$\mathbf{J} = \begin{bmatrix} \mathbb{I}_d & \mathbf{0}_{d\times(D-d)} \\ \frac{\partial \mathbf{y}_{d+1:D}}{\partial \mathbf{x}_{1:d}} & \operatorname{diag}\left(\exp\left(s\left(\mathbf{x}_{1:d}\right)\right)\right) \end{bmatrix}$$

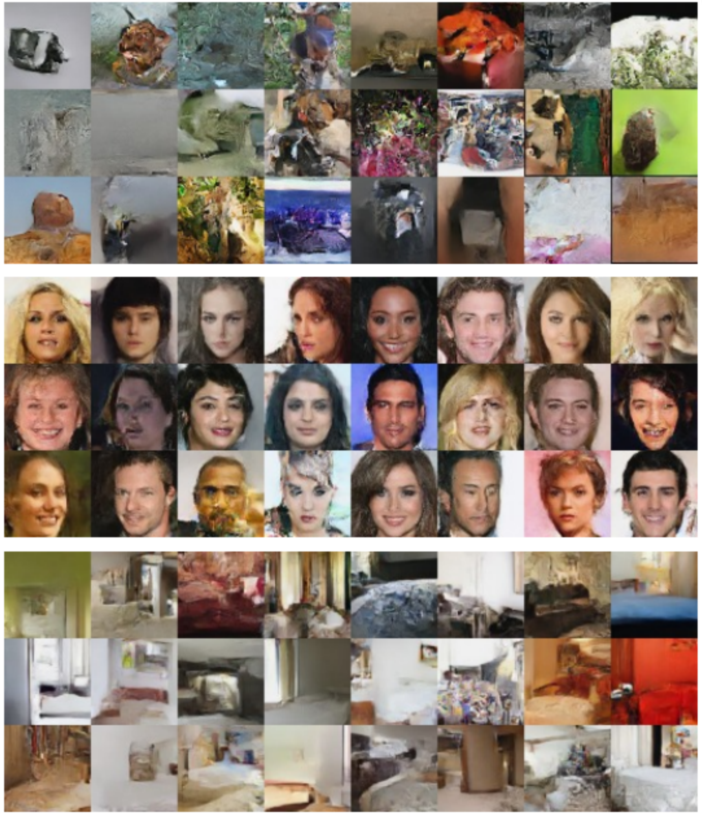
$$\det(\mathbf{J}) = \prod_{j=1}^{D-d} \exp\left(s\left(\mathbf{x}_{1:d}\right)\right)_j = \exp\left(\sum_{j=1}^{D-d} s\left(\mathbf{x}_{1:d}\right)_j\right)$$

Dinh, L., Sohl-Dickstein, J., & Bengio, S. (2016). Density estimation using real nvp. arXiv preprint arXiv:1605.08803.

RESULTS

RESULTS

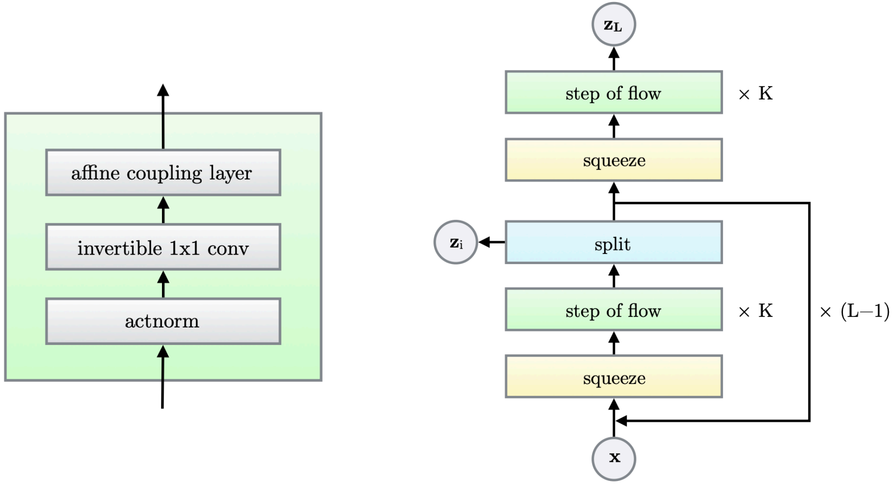




GLOW: REALNVP WITH 1X1 CONVOLUTIONS

A model contains ~1000 convolutions.

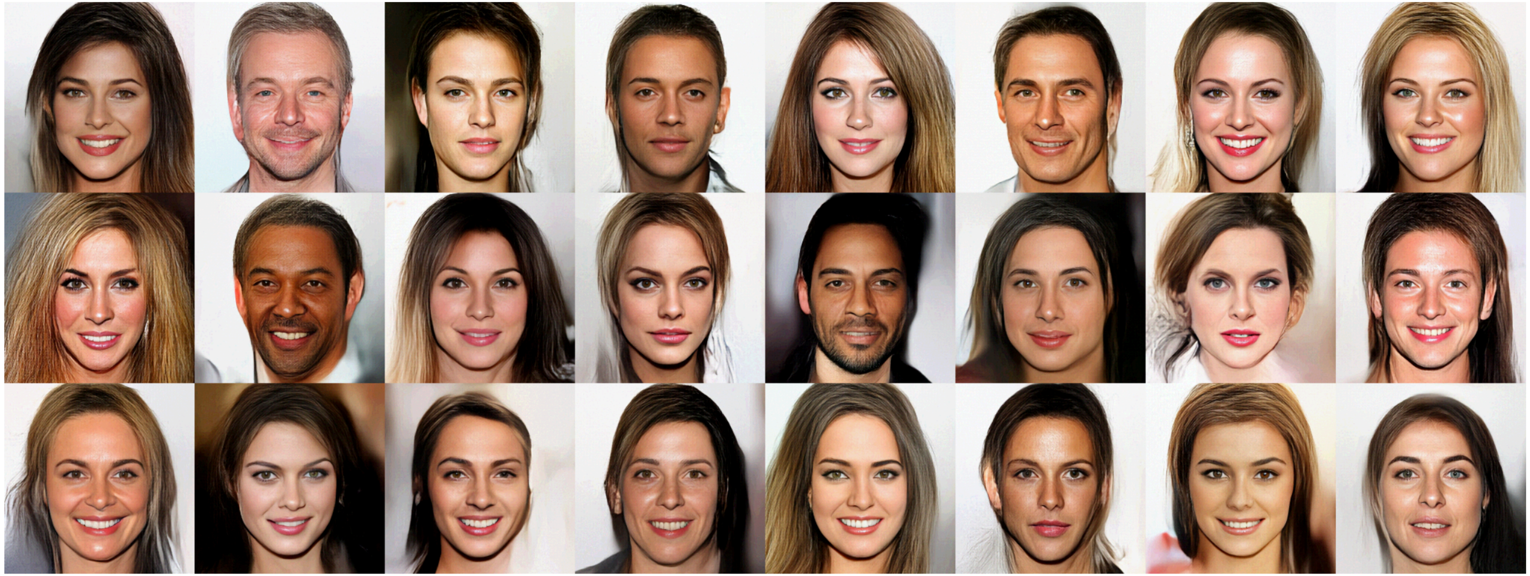
A new component: 1x1 convolution instead of a permutation matrix.



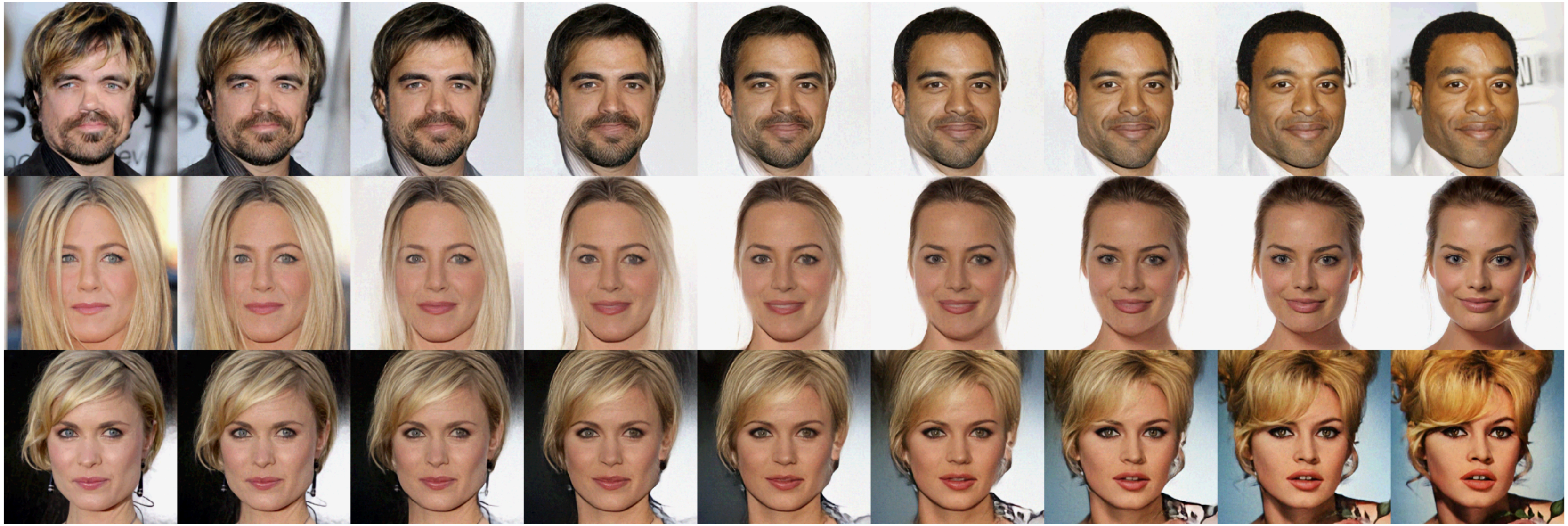
(a) One step of our flow. (b) Multi-scale architecture (Dinh et al., 2016).

Kingma, D. P., & Dhariwal, P. (2018). Glow: Generative flow with invertible 1x1 convolutions. *NeurIPS 2018*

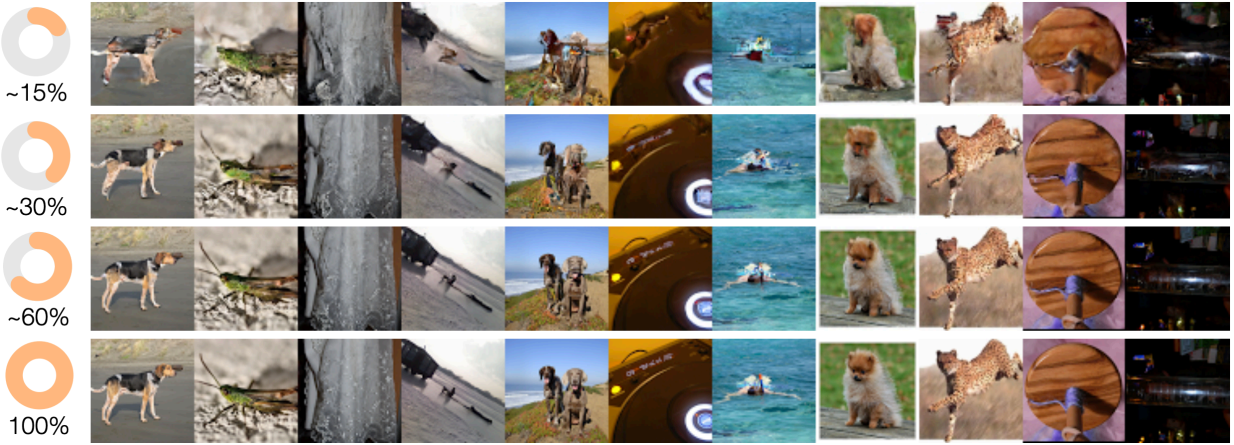
GLOW: SAMPLES



GLOW: LATENT INTERPOLATION



INTEGER DISCRETE FLOW: NO NEED TO CALCULATE JACOBIAN!



Hoogeboom, E., Peters, J. W., Berg, R. V. D., & Welling, M. (2019). Integer Discrete Flows and Lossless Compression. *NeurIPS 2019*

Future directions

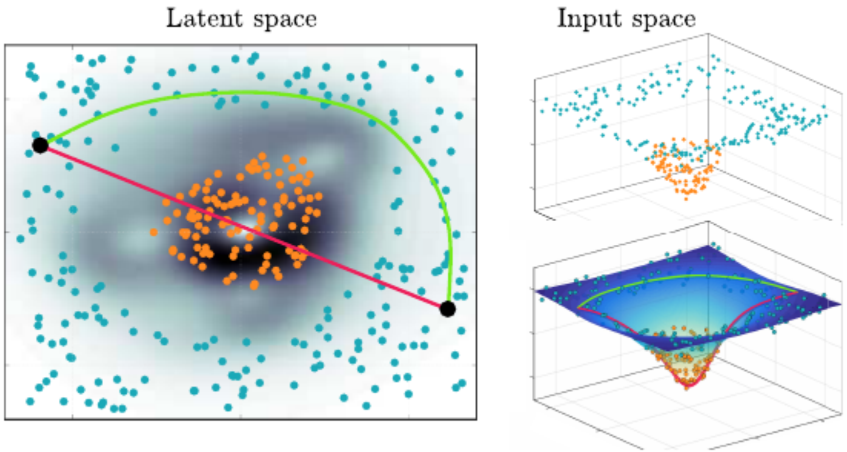
BLURRINESS AND SAMPLING IN VAES

How to avoid sampling from **holes**?

Should we follow **geodesics in the latent space**?

How to use **geometry** of the latent space to build better **decoders**?

How to build **temporal decoders**?



VU

COMPRESSION AND VAES

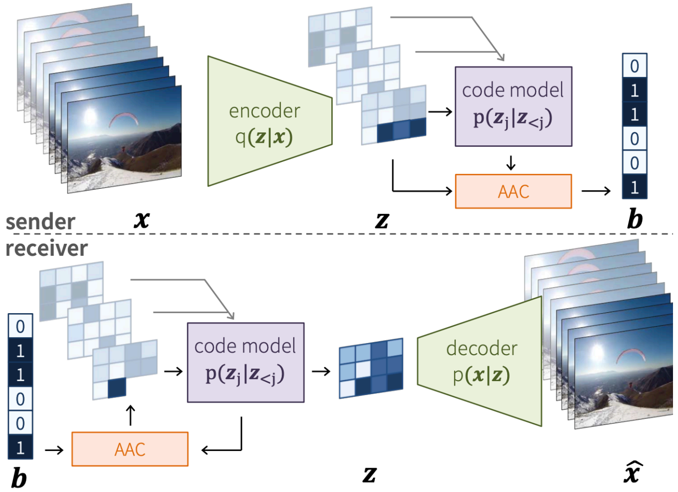
Taking a **deterministic & discrete encoder** allows to simplify the objective.

$$\mathrm{RE}(x|z) - \mathrm{H}[q(z|x)] - \mathrm{CE}[q(z)||p(z)]$$
$$= \mathrm{RE}(x|z) - \mathrm{CE}[q(z)||p(z)]$$

It is important to learn a **powerful prior**. This is challenging!

Is it **easier** to learn a prior with **temporal dependencies**?

Can we alleviate some dependencies by using **hypernets**?



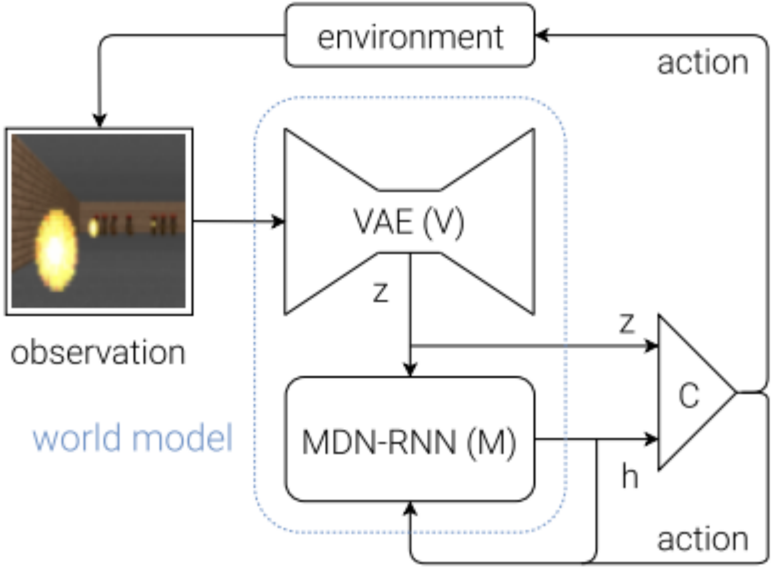
Habibian, A., van Rozendaal, T., Tomczak, J.M., & Cohen, T.S. (2019), Video Compression with Rate-Distortion Autoencoders, ICCV 2019

ACTIVE LEARNING/RL AND VAES

Using **latent representation** to navigate and/or quantify uncertainty.

Formulating **policies** in the latent space entirely.

Do we need a better notion of **sequential dependencies**?



Ha, D., & Schmidhuber, J. (2018). World models. *arXiv preprint arXiv:1803.10122*.

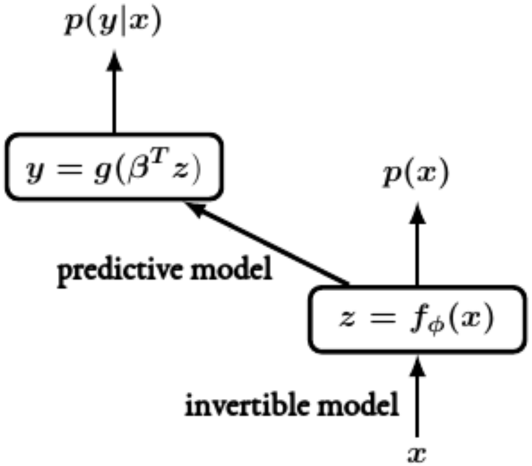
HYBRID AND FLOW-BASED MODELS

We need a **better understanding** of the latent space.

Joining an **invertible model** (flow-based model) with a **predictive model**.

Isn't this model an **overkill**?

How would it work in the **multi-modal learning** scenario?



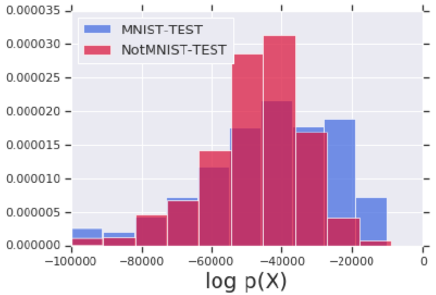
Nalisnick, E., et al. (2019). Hybrid models with deep and invertible features. *arXiv preprint arXiv:1902.02767*.

HYBRID MODELS AND OOD SAMPLE

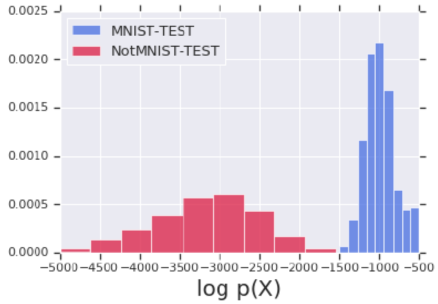
Going back to first slides, we need a good notion of **p(x)**.

Distinguishing **out-of-distribution (OOD)** samples is very important.

Crucial for **decision making**, **outlier detection**, **policy learning**…



(a) Discriminative Model ($\lambda = 0$)



(b) Hybrid Model

Nalisnick, Eric, et al. "Hybrid models with deep and invertible features." *arXiv preprint arXiv:1902.02767* (2019).

Thank you!

Webpage:

https://jmtomczak.github.io/

Code on github:

https://github.com/jmtomczak/

Contact:

jmk.tomczak@gmail.com